REMARKS

Claims 1-21 are pending. Claims 1, 8, and 17-21 have been amended. No new matter has been introduced. Reexamination and reconsideration of the present application are respectfully requested.

In the October 3, 2002 Office Action, the Examiner rejected claims 1-21. The Examiner rejected claims 1-3, 6, and 7 under 35 U.S.C. § 103(a) as being obvious over U.S. Patent No. 6,434,618 to Cohen (the Cohen reference), in view of U.S. Patent No. 6,424,621 to Ramaswamy et al. (the Ramaswamy reference). The Examiner rejected claims 4 and 5 under 35 U.S.C. § 103(a) as being obvious over the Cohen reference, in view of the Ramaswamy reference, and further in view of U.S. Patent No. 5,742,607 to Beighe et al. (the Beighe reference). The Examiner rejected claims 8-16 for the reasons as set forth with respect to claims 1-7 above. The Examiner rejected claims 17-21 for the reasons as set forth with respect to claims 1-7 above. These rejections are respectfully traversed.

The present invention relates to a computer system that allows proprietary forwarding elements to interoperate with standard control elements in an open network architecture. The computer system includes a forwarding element that is adapted to perform data forwarding functions in a computer network. A control element is adapted to perform network signaling and control functions in the computer network. The control element is adapted to generate a uniform standardized data set for configuring the forwarding element. An interconnecting element operatively connects the forwarding element to the control element. A forwarding element plugin is integrated with the control element for receiving the uniform standardized data set from the control

element, translating the uniform standardized data set into a proprietary specialized data set to the forwarding element, and transmitting the proprietary specialized data set to the forwarding element to configure the forwarding element. The forwarding element utilizes the proprietary specialized data set to configure the forwarding element for performing data forwarding in the computer network.

Independent claim 1, as amended, recites:

> a forwarding element adapted to perform data forwarding in a computer network;
>
> a control element adapted to perform network signaling and control in the computer network, wherein *the control element is adapted to generate a uniform standardized data set for configuring the forwarding element and is external to the forwarding element*;
>
> an interconnecting element operatively connecting the forwarding element to the control element; and
>
> a *forwarding element plugin integrated with the control element for receiving the uniform standardized data set from the control element, translating the uniform standardized data set into a proprietary specialized data set to the forwarding element, and transmitting the proprietary specialized data set to the forwarding element to configure the forwarding element*, wherein the forwarding element utilizes the proprietary specialized data set to configure the forwarding element for performing data forwarding in the computer network.

The Cohen reference is directed to a programmable network element that operates on packet traffic flowing through the element in accordance with a gateway

program that is dynamically uploaded into the network element or unloaded from it via a mechanism separate from the actual packet traffic as the element operates. The programmable network element simultaneously operates on plural packet flows with different or the same programs being applied to each flow. A dispatcher provides a packet filter with a set of rules provided by one or more of the dynamically loaded and invoked programs. These rules define, for each program, the characteristics of those packets flowing through the network element that are to be operated upon in some manner. A packet that flows from the network through the filter and satisfies one or more of such rules is sent by the packet filter to the dispatcher. The dispatcher, in accordance with one of the programs, either sends the packet to the program for manipulation by the program itself, or manipulates the packet itself in a manner instructed by the program. The processed packet is sent back through the filter to the network for routing to its destination.

The Cohen reference does not disclose, teach, or suggest the computer system of independent claim 1, as amended. As already acknowledged by the Examiner, the Cohen reference "does not expressly disclose a control element adapted to perform network signaling and control in the computer network, wherein the control element is adapted to generate a standardized data set for configuring the forwarding element" (Office Action, page 3, lines 4-7). Moreover, unlike independent claim 1, as amended, the Cohen reference makes no mention of a *forwarding element plugin integrated with the control element for receiving the uniform standardized data set from the control element, translating the uniform standardized data set into a proprietary specialized data set to the forwarding element, and transmitting the proprietary specialized data set*

to the forwarding element to configure the forwarding element. The Cohen reference only discloses a programmable network element that operates on standardized packet traffic passing through the network element, and may be utilized in a network as a router, or placed at an edge of a network between one or more LANs and the rest of the network, or at an edge between an internal network and an external network (col. 2, lines 5-36; and Figs. 1-4).

The Ramaswamy reference does not make up for the deficiencies of the Cohen reference. The Ramaswamy reference is directed to a data packet switching system having a plurality of network interfaces each adapted to be coupled to respective external networks for receiving and sending data packets to and from the external networks via a particular communication protocol. The data packet switching system further includes a plurality of symmetrical processors, including a first processor providing a control processor and remaining ones of the processors each providing data packet switching processors. The data packet switching processors are coupled to the plurality of network interfaces. The control processor further includes a user portion and an operating system portion. The operating system portion is provided with a pseudo-network driver that appears to be a network interface to user application programs operating on the user portion of the control processor. A memory space is shared by the control processor and the data packet switching processors. The data packet switching processors route an incoming data packet directed to a user application program to the memory space. The pseudo-network driver retrieves the incoming data packet from the shared memory space and provides the data packet to the user application program.

The Ramaswamy reference does not disclose, teach, or suggest the computer system of independent claim 1, as amended. Unlike independent claim 1, as amended, the Ramaswamy reference does not disclose a control element adapted to perform network signaling and control in the computer network, wherein *the control element is adapted to generate a uniform standardized data set for configuring the forwarding element and is external to the forwarding element*, and a *forwarding element plugin integrated with the control element for receiving the uniform standardized data set from the control element, translating the uniform standardized data set into a proprietary specialized data set to the forwarding element, and transmitting the proprietary specialized data set to the forwarding element to configure the forwarding element.* The Ramaswamy reference only shows that a resource manager of a control processor is adapted to receive raw data from back-end application servers indicating their present load status (col. 7, lines 1-24; and Fig. 4), and the switching processors of the load balancing and switching system may be viewed as entirely separate logical networking end points even though they reside within a single physical device (col. 10, lines 20-36; and Fig. 4).

The Ramaswamy reference does not teach a *control element* that generates a *uniform standardized data set for configuring the forwarding element and is external to the forwarding element*, as recited in independent claim 1, such that the hardware-specific functionality of the data forwarding element is available to application programmers in a uniform, standardized, hardware-independent manner. The Ramaswamy reference also does not teach a *forwarding element plugin integrated with the control element for receiving the uniform standardized data set from the control*

element, translating the uniform standardized data set into a proprietary specialized data set to the forwarding element, and transmitting the proprietary specialized data set to the forwarding element to configure the forwarding element, as recited in independent claim 1, as amended, such that the forwarding element platform-specific functionality is available to the control element and to independent software vendors without revealing the hardware-specific details of the forwarding element. Therefore, the forwarding element plugin allows an independent hardware vendor to provide a software module that captures the hardware-specific functionality of the forwarding element in a uniform, standardized, hardware-independent manner via the *uniform, standardized data set*, which is then *translated* into the *proprietary specialized data set to the forwarding element*, as recited in independent claim 1, as amended, which is not mentioned at all in the Ramaswamy reference.

The Beighe reference does not make up for the deficiencies of the Cohen reference and the Ramaswamy reference. The Beighe reference is directed to a method and apparatus for controlling two way communication via disparate physical media. A computer includes a central processor, a forward channel interface, a return channel interface, and a main memory, each being coupled to a bus. The forward channel interface is further coupled to interrupt the central processor and coupled to receive a packet from a forward channel. The main memory contains an interrupt service routine having a first set of code for passing the packet to a routine for decapsulating the packet, and a second set of code for passing a second packet to the return channel interface. The method includes a computer transferring a packet from a forward channel interface to a main memory. The central processor analyzes the

packet to determine if the packet is a data packet or a network management packet. If the packet is a network management packet, the central processor creates a response packet and passes the response packet to a return channel interface.

The Beighe reference does not disclose, teach, or suggest the computer system of independent claim 1, as amended. Unlike independent claim 1, as amended, the Beighe reference does not make any mention of a control element adapted to perform network signaling and control in the computer network, wherein *the control element is adapted to generate a uniform standardized data set for configuring the forwarding element and is external to the forwarding element*, and a *forwarding element plugin integrated with the control element for receiving the uniform standardized data set from the control element, translating the uniform standardized data set into a proprietary specialized data set to the forwarding element, and transmitting the proprietary specialized data set to the forwarding element to configure the forwarding element.* The Beighe reference only shows a system for controlling two-way communications having an interrupt service routine that handles the decapsulation of data packets (col. 2, lines 24-48; and Fig. 3). Accordingly, applicants respectfully submit that independent claim 1, as amended, distinguishes over the above-cited references.

Independent claims 8 and 17, both as amended, recite limitations similar to independent claim 1, as amended. Claims 2-7 all directly depend from independent claim 1, as amended. Claims 9-16 all directly depend from independent claim 8, as amended. Claims 18-21, all as amended, all directly depend from independent claim 17, as amended. Accordingly, applicants respectfully submit that claims 2-21 distinguish over the above-cited references for the reasons set forth above with respect

to independent claim 1, as amended.

Applicants believe that the foregoing amendments place the application in condition for allowance, and a favorable action is respectfully requested. If for any reason the Examiner finds the application other than in condition for allowance, the Examiner is requested to call either of the undersigned attorneys at the Los Angeles telephone number (213) 488-7100 to discuss the steps necessary for placing the application in condition for allowance should the Examiner believe that such a telephone conference would advance prosecution of the application.

Respectfully submitted,

PILLSBURY WINTHROP LLP

Date: December 16, 2002

By: _____
Eric S. Chen
Registration No. 43,542
Attorney for Applicant(s)

Date: December 16, 2002

By: _____
Roger R. Wise
Registration No. 31,204
Attorney For Applicant(s)

725 South Figueroa Street, Suite 2800
Los Angeles, CA 90017-5406
Telephone: (213) 488-7100
Facsimile: (213) 629-1033

APPENDIX

VERSION WITH MARKINGS TO SHOW CHANGES MADE

IN THE SPECIFICATION:

Please amend the paragraph starting at page 9, line 18 as follows:

The network services API 240 allows the horizontal open networking architecture 200 and platform to be visible to external parties, such as third-party independent software vendors. The network services API 240 makes the hardware-specific functionality of the data forwarding elements 210 (such as switches, routers, and network interface cards (NICs)) available to the application programmers in a uniform, hardware-independent manner. Third party independent software vendors that write network-aware applications, such as VoIP (voice-over internet protocol) gateways, intrusion detection, application-specific proxies, and VPN (virtual private network) servers, use the network service API 240 to control/modify the behavior of the data forwarding path in both the data and control planes. For example, in the case of a L3/L4 switch with the ability to filter packets based on pre-specified packet filters, an H.323 (H.323, Packet-Based Multimedia Communication Systems, International Telecommunication Union, February 1998) proxy uses the network services API 240 to direct the forwarding elements 210 to intercept and forward H.323-related packets to itself for further processing. However, in the case of a L4/L7 switch with the capability of stateful inspection of data packets at wire-speed, an intrusion detection application could install a policy rule in the forwarding element 210 that specifies the classification filter and the associated actions for examining session state and identifying an "intrusion signature". In the horizontal open networking architecture 200 according to

an embodiment of the present invention, the control element 230 hosts the network services API 240 implementation and hides the details of translating the network services API 240 calls to appropriate message passing and invocations of hardware-specific calls at the forwarding element 210.

Please amend the paragraph starting at page 14, line 7 as follows:

The connect API 220 itself includes the basic communication primitives along with methods for examining and manipulating the status of the underlying transport in a transport independent manner. The connect API 220 may be implemented over a variety of transports, such as PCI (peripheral component interconnect), [NGIO] input/output backplane, Ethernet, or ATM (asynchronous transfer mode). Depending on the transport, a transport-specific module specifies exactly how the connect API UDPs (user datagram protocols) are encapsulated on specific interconnect technologies, as well as how to deal with normal and exception conditions in the operation of the interconnect technology. Two examples of transports include PCI and IP. The IP transport is preferably used over the Ethernet or ATM, and can be implemented using either UDP (user datagram protocol) or TCP (transmission control protocol) as transport protocols. The use of these protocols, however, does not preclude the possibility of using native ATM or Ethernet transport for implementation of the connect API 220.

Please amend the paragraph starting at page 16, line 13 as follows:

The forwarding element-specific configuration BLOB (specialized data set) 495,

contains the forwarding element-specific invocations of functionality that is specific to the proprietary forwarding element 210. By utilizing the forwarding element-specific plugin 490 to translate a standardized data set into a specialized data set, the proprietary forwarding element 210 implementations need not be exposed. Therefore, the proprietary design and architecture information are kept confidential. Because the forwarding element-specific plugin 490 is in binary form, the transformation process from the standardized data set to the specialized data set is essentially hidden from [everyone] other components, thus protecting the proprietary forwarding element 210 implementation. A tremendous amount of complex reverse engineering would be required in order to determine the translation process from the forwarding element specific plugin DLL file.

Please amend the paragraph starting at page 17, line 1 as follows:

Once the forwarding element-specific configuration BLOB 495 is generated, it is preferably passed back to the opaque forwarding element plugin API 480, which then transmits the BLOB 495 to the proprietary forwarding element 210. The BLOB 495 is preferably passed through the open forwarding element/control element interconnect 220, then to the abstract forwarding element API 440, and then finally to the BLOB decapsulator 430. The BLOB decapsulator 430 takes the BLOB 495 and "decapsulates" it — which is a very [lightweight] simple computational operation — and passes the decapsulated BLOB data directly to the device-specific forwarding element interface 420. The device-specific forwarding element interface 420 takes the information from the decapsulated BLOB data to configure the forwarding element

software and hardware 410 to properly operate the proprietary forwarding element 210. In this manner, confidential information about the proprietary forwarding element's architecture that is present in the device-specific forwarding element interface 420 is never exposed to the standard control element 230, allowing independent hardware vendors of the forwarding elements to protect their intellectual property.

IN THE CLAIMS:

Please amend claims 1, 8, and 17-21 as follows:

1. (Amended) A computer system comprising:

a forwarding element adapted to perform data forwarding in a computer network;

a control element adapted to perform network signaling and control in the computer network, wherein the control element is adapted to generate a uniform standardized data set for configuring the forwarding element and is external to the forwarding element;

an interconnecting element operatively connecting the forwarding element to the control element; and

a forwarding element plugin integrated with the control element for receiving the uniform standardized data set from the control element, translating the uniform standardized data set into a proprietary specialized data set to the forwarding element, and transmitting the proprietary specialized data set to the forwarding element to configure the forwarding element, wherein the forwarding element utilizes the proprietary specialized data set to configure the forwarding

element for performing data forwarding in the computer network.

8. (Amended) A method of configuring a computer device, the method comprising:

generating a uniform standardized data set by a control element for configuring a forwarding element, wherein the control element is external to the forwarding element;

transmitting the uniform standardized data set from the control element to a forwarding element plugin integrated with the control element;

translating the uniform standardized data set into a proprietary specialized data set to the forwarding element; and

transmitting the proprietary specialized data set to the forwarding element for configuring the forwarding element.

17. (Amended) [A forwarding element plugin software program comprising:

a computer-readable medium; and

a computer-readable program code, stored on the computer-readable medium, adapted to be integrated with a control element for configuring a forwarding element, the computer-readable program code performing,]

An article comprising a machine-readable medium storing instructions when executed by a processor, the instructions,

receiving a uniform standardized data set for configuring the forwarding element generated by the control element, wherein the control element is external to the forwarding element,

translating the uniform standardized data set into a proprietary specialized data set to the forwarding element, and

transmitting the proprietary specialized data set to the forwarding element for configuring the forwarding element.

18. The [forwarding element plugin software program] article according to claim 17, wherein the [computer-readable program code] instructions further perform[s]:

receiving the uniform standardized data set from an opaque forwarding element plugin; and

transmitting the proprietary specialized data set to the opaque forwarding element plugin.

19. The [forwarding element plugin software program] article according to claim 17, wherein the [computer-readable program code] instructions further perform[s]: encrypting the proprietary specialized data set before transmission to the forwarding element.

20. The [forwarding element plugin software program] article according to claim 17, wherein the proprietary specialized data set [is] includes a binary large object.

21. The [forwarding element plugin software program] article according to claim 17, wherein the [computer-readable program code is] instructions include a dynamic link library.